SCE221 – Verificação, Validação e Teste
SCE702 – Teste e Inspeção de Software

Inspeção de Software

Profa. Ellen Francine Barbosa
Profa. Simone do Rocio Senger de Souza
{francine, srocio}@icmc.usp.br

Instituto de Ciências Matemáticas e de Computação — ICMC/USP

Roteiro

- Revisões de Software
- Revisões Técnicas Formais
- Reunião de Revisão Técnica
- Inspeção de Software
- Inspeção em DR
- Inspeção em Código-Fonte
- Conclusão

Aula Anterior...

- Qualidade de software.
- V&V envolvem atividades de análise estática e de análise dinâmica.
- Análise Estática
  - Não requerem a execução propriamente dita do produto.
  - Podem ser aplicadas em qualquer produto intermediário do processo de desenvolvimento.
    - Documento de requisitos, diagramas de projeto, código-fonte, planos de teste, ...
  - As revisões são o exemplo mais clássico de análise estática.

Revisões de Software I

- Meio efetivo para melhorar a qualidade de software.
  - Filtro para o processo de Engenharia de Software.
- Podem ser aplicadas em vários pontos durante o desenvolvimento do software.
- Maneira de usar a diversidade de um grupo de pessoas para:
  - Apontar melhorias necessárias ao produto.
  - Confirmar as partes de um produto em que uma melhoria não é desejada ou não é necessária.
  - Realizar um trabalho técnico de qualidade mais uniforme de forma a torná-lo mais administrável.

Objetivo das Revisões de Software I

- Encontrar erros durante o processo de desenvolvimento, de forma que eles não se transformem em defeitos depois da entrega do software.
- Descoberta precoce dos erros.
  - Melhoria da qualidade já nas primeiras fases do processo de desenvolvimento.
  - Aumento da produtividade e diminuição dos custos.
    - Erros são detectados quando sua correção é mais barata.

Objetivo das Revisões de Software II

- É uma oportunidade de treinamento.
  - Aprender por experiência.
  - Participantes aprendem as razões e padrões em descobrir erros.
  - Participantes aprendem bons padrões de desenvolvimento de software.
- Com o decorrer do tempo....
  - A revisão auxilia os participantes a desenvolver produtos mais fáceis de entender e de manter.

Tipos de Revisões

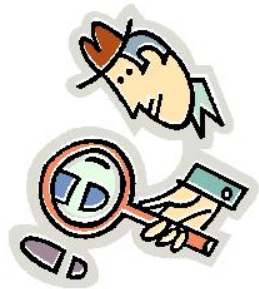
- Discussão informal de um problema técnico.
- Apresentação do projeto de software para uma audiência de clientes, administradores e pessoal técnico.
- Revisões Técnicas Formais (RTF), as quais incluem avaliações técnicas do software realizadas em pequenos grupos.
  - Inspeção
  - Walkthrough
  - Peer-Review

Revisões Técnicas Formais

- Métodos de RTF:
  - Inspeção
  - Walkthrough
  - Peer-Review

Inspeção

- Método de análise estática para verificar a qualidade de um produto de software.
- Como detectar defeitos?
  - Lendo o documento.
  - Entendendo o que o documento descreve.
  - Checando as propriedades de qualidade requeridas.



Walkthrough

- Procedimento similar ao procedimento para condução de uma inspeção.
- A diferença fundamental está na maneira como a sessão de revisão é conduzida.
  - Em vez de ler o programa ou checar os erros por meio de um *checklist*, os participantes simulam sua execução.
  - Papel adicional: testador.
    - Elaborar um pequeno conjunto de casos de teste (em papel).
    - Monitorar e controlar os resultados obtidos.

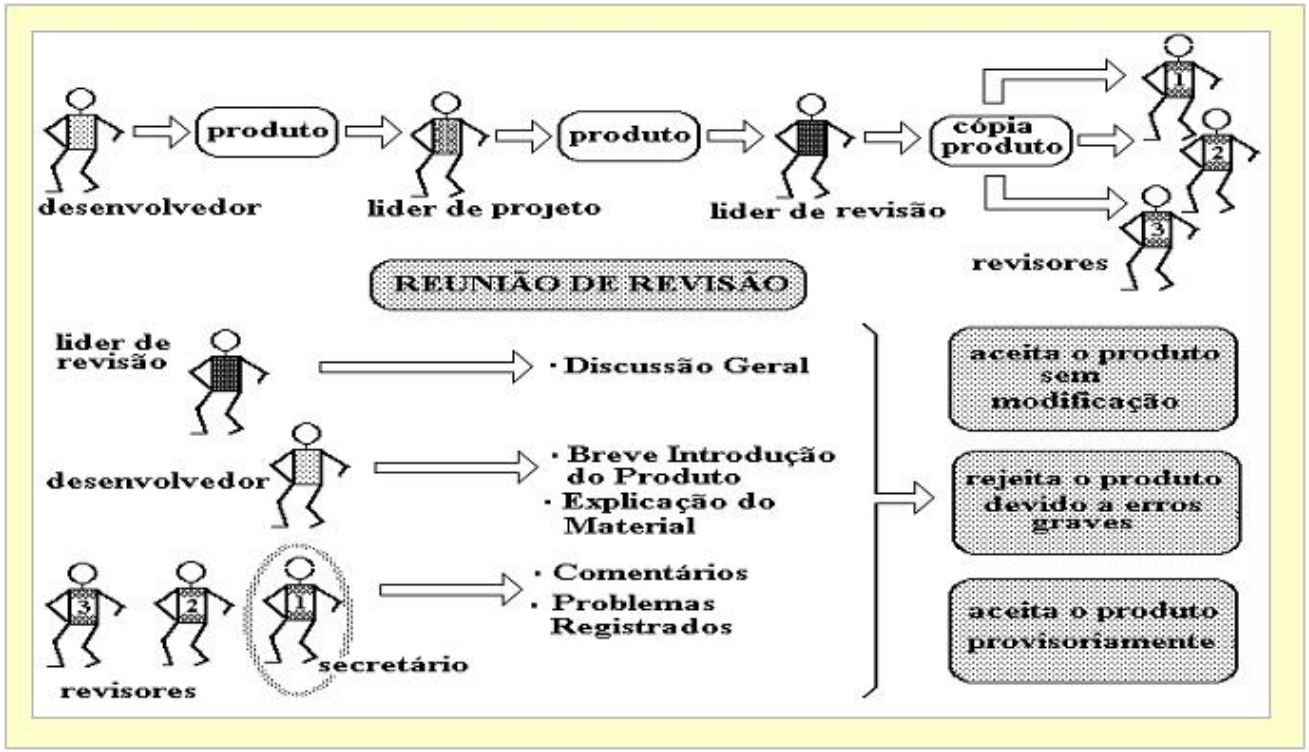
Peer-Review

- Conduzida por pares de programadores.
  - Mesmo nível de conhecimento.
- Aplicada ao código.
- Reuniões com duração de 1 a 2 horas.
  - Somente um programa ou parte dele (rotinas) deve ser revisado.
- Resultados são publicados em um relatório informal.
  - Não faz parte da documentação oficial do projeto.

Reunião de Revisão Técnica I

- Independentemente do formato da RTF, toda reunião de revisão deve seguir as seguintes recomendações:
  - Envolver de 3 a 5 pessoas.
  - Deve haver uma preparação para a reunião.
    - A preparação não deve exigir mais de 2 horas de trabalho de cada pessoa.
  - A reunião deve durar menos de 2 horas.
  - Deve-se focalizar uma parte específica do software.
    - Maior probabilidade de descobrir erros.

Reunião de Revisão Técnica II



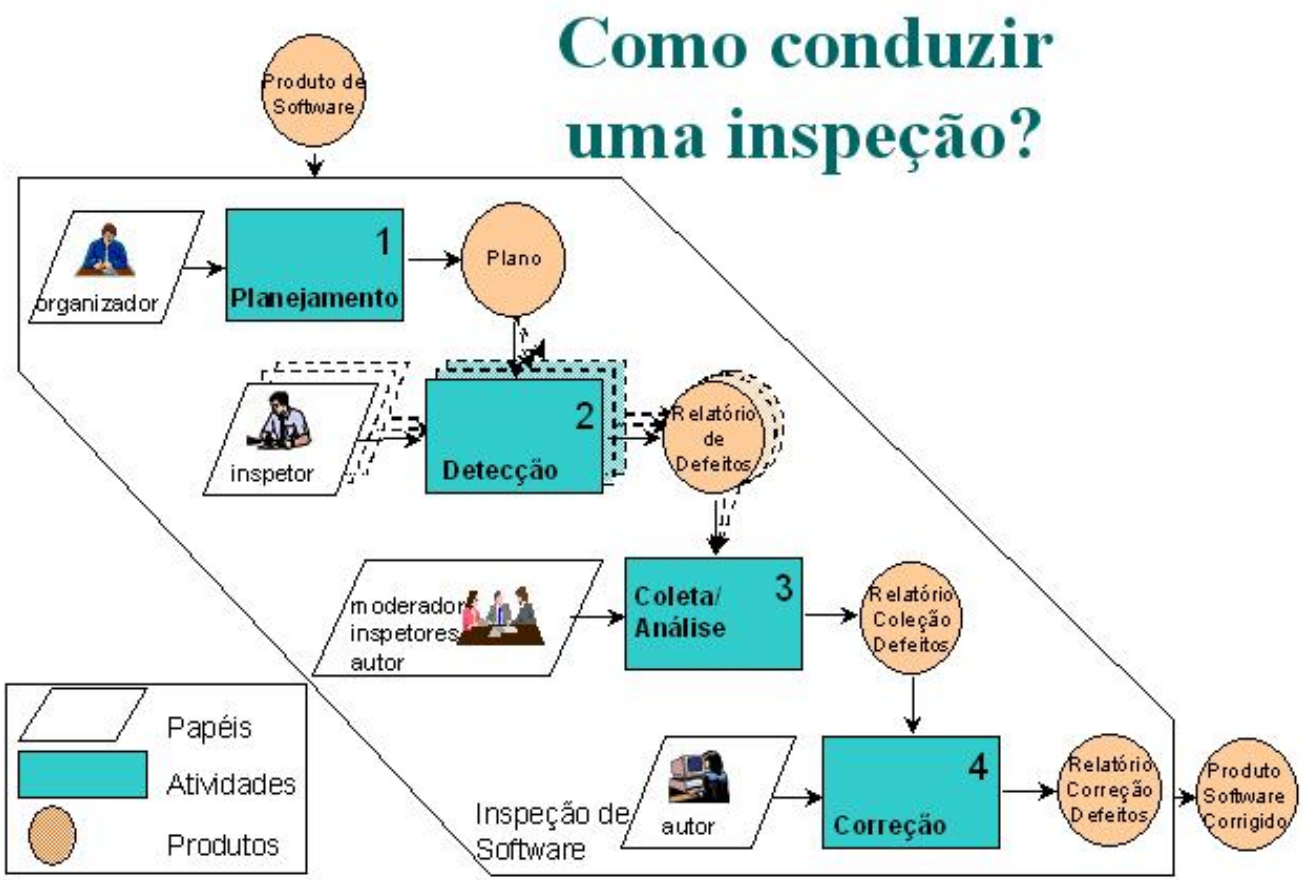
Diretrizes para Revisão

- Revise o produto, não o produtor.
- Fixe e mantenha uma agenda.
- Limite o debate e a refutação.
- Relacione as áreas problemáticas.
- Faça anotações por escrito.
- Limite o número de participantes e insista em uma preparação antecipada.
- Desenvolva uma lista de conferência (checklist) para cada produto que provavelmente será revisto.
- Atribua recursos e uma programação de tempo para as revisões.
- Realize um treinamento significativo para todos os revisores.
- Reveja suas antigas revisões.

Inspeção de Software

- Método de análise estática para verificar a qualidade de um produto de software.
- Pode-se inspecionar tanto produtos de software como também projetos de software.
  - Diferencial está na seleção dos aspectos que devem ser considerados durante a revisão.
- **Inspeção em Documentos de Requisitos**.
- **Inspeção em Código-fonte**.

Etapas da Inspeção



Vantagens da Inspeção

- Detecção antecipada de defeitos (inspeção de requisitos).
- Aprende-se pela experiência.
  - Participantes aprendem os padrões e o raciocínio utilizado na detecção de defeitos.
  - Participantes aprendem bons padrões de desenvolvimento.
- A longo prazo...
  - A inspeção convence os participantes a desenvolverem produtos mais compreensíveis e mais fáceis de manter.

As inspeções ajudam a integrar o processo de **prevenção** de defeitos com o processo de **detecção** de defeitos.

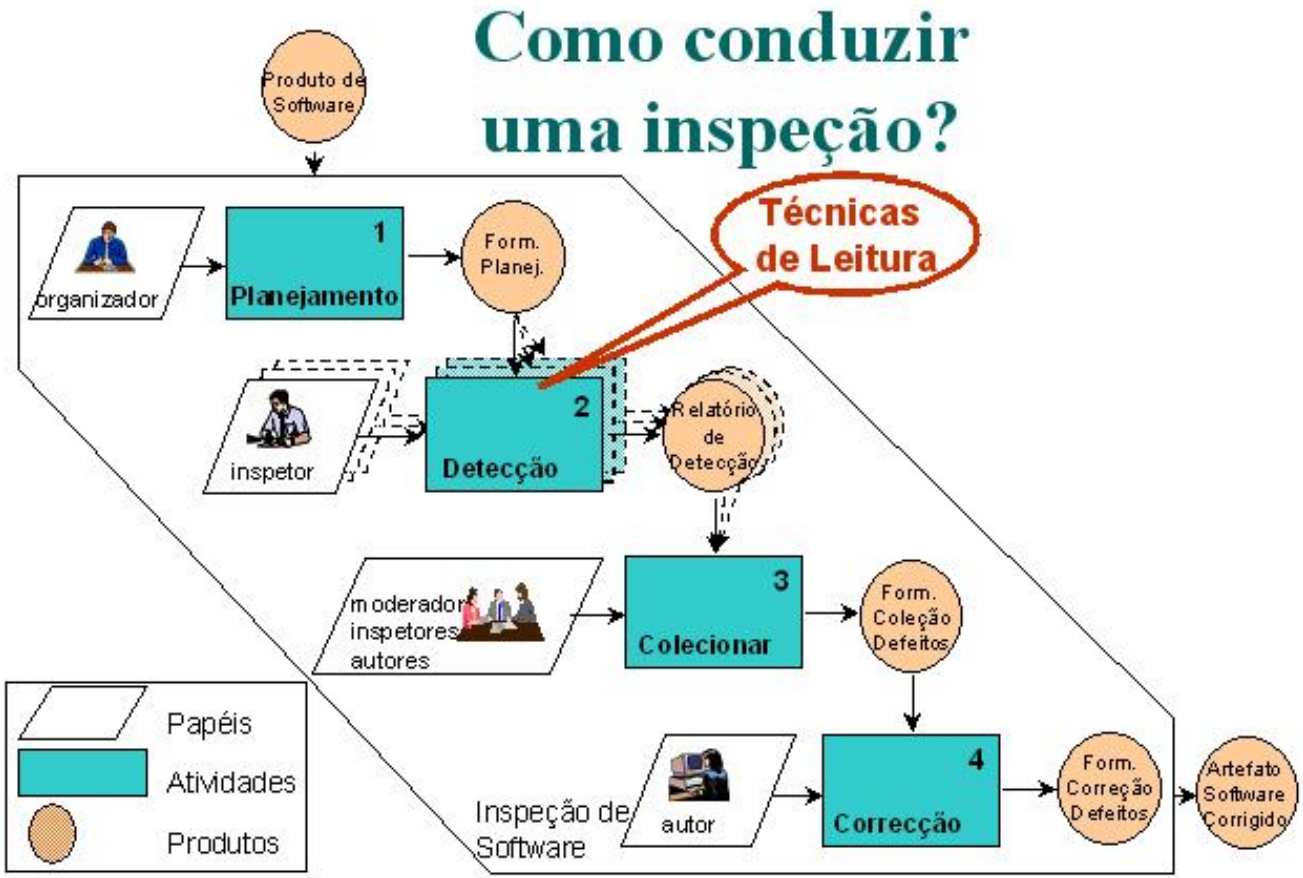
Técnicas de Leitura para Inspeção I

- Como detectar defeitos?
  - Lendo o documento.
  - Entendendo o que o documento descreve.
  - Verificando as propriedades de qualidade requeridas.
- Problema:
  - **Em geral não se sabe como fazer a leitura de um documento!!!**
- Razão:
  - Em geral, os desenvolvedores aprendem a escrever documento de requisitos, código, projeto, mas não aprendem a fazer uma leitura adequada dos mesmos.

Técnicas de Leitura para Inspeção II

- Solução:
  - Fornecer **técnicas de leitura** bem definidas.
- Benefícios:
  - Aumenta a relação custo/benefício das inspeções.
  - Fornece modelos para escrever documentos com maior qualidade.
  - Reduz a subjetividade nos resultados da inspeção.

Técnicas de Leitura para Inspeção III



Técnicas de Leitura para Inspeção IV

- O que é uma técnica de leitura?
  - Conjunto de instruções fornecido ao revisor dizendo como ler e o quê procurar no produto de software.
- Algumas técnicas de leitura:
  - Ad-hoc
  - Checklist
  - Leitura Baseada em Perspectiva (PBR)
  - Stepwise Abstraction

Ad-hoc

- Os revisores não utilizam nenhuma técnica sistemática de leitura.
- Cada revisor adota sua própria maneira de ler o documento.
- Desvantagens:
  - Depende da experiência do revisor.
  - Não é repetível.
  - Não é passível de melhoria pois não existe um procedimento a ser seguido.

Checklist

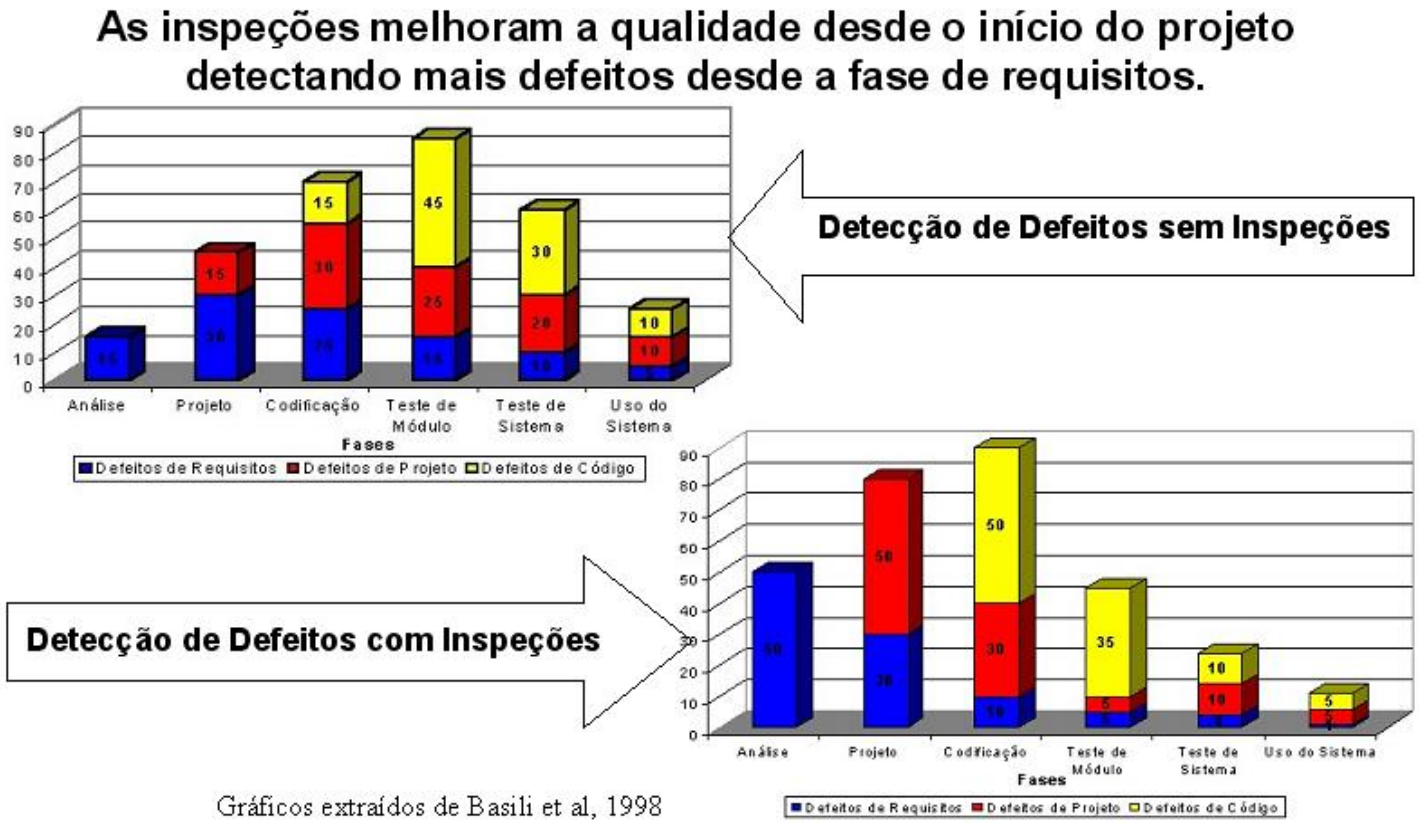
- Técnica que fornece diretrizes para ajudar o revisor alcançar os objetivos de uma atividade de revisão formal.
- Similar ao ad-hoc, mas cada revisor recebe um **checklist**.
  - Os itens do checklist capturam lições importantes que foram aprendidas em inspeções anteriores no ambiente de desenvolvimento.
  - Itens do checklist podem explorar defeitos característicos, priorizar defeitos diferentes e estabelecer questões que ajudam o revisor a encontrar defeitos.

Inspeção em Documento de Requisitos

- Benefícios da inspeção em Documento de Requisitos:
  - Detecção antecipada de defeitos.
  - Produtividade e diminuição do custo.

Inspeção em Documento de Requisitos

- Detecção antecipada de defeitos.



Gráficos extraídos de Basili et al, 1998

Inspeção em Documento de Requisitos

- Produtividade e diminuição do custo.

Inspeções melhoram produtividade por acharem defeitos quando eles são mais baratos para corrigir.

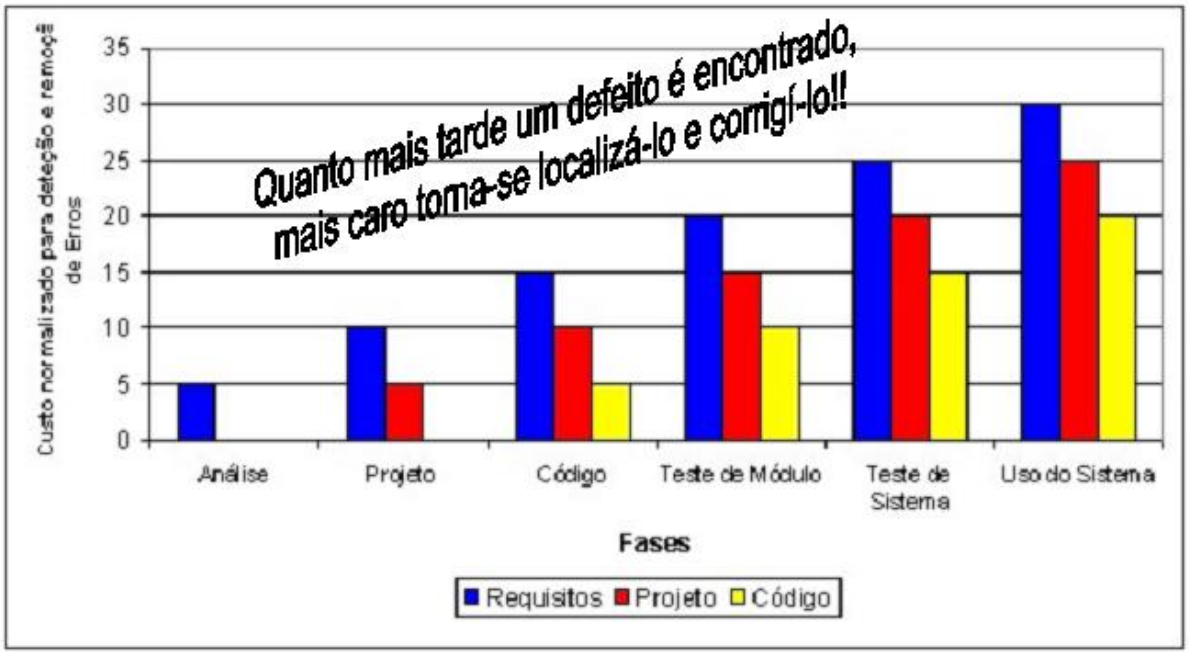
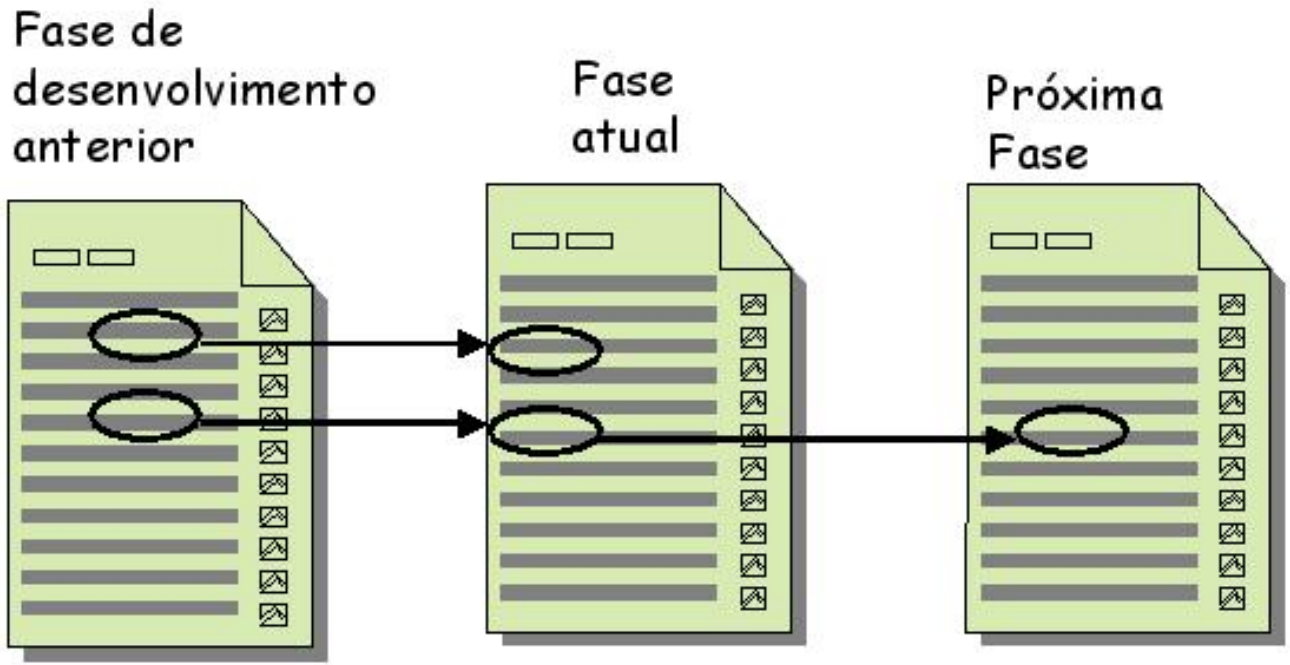


Gráfico extraído de Basili et al, 1998

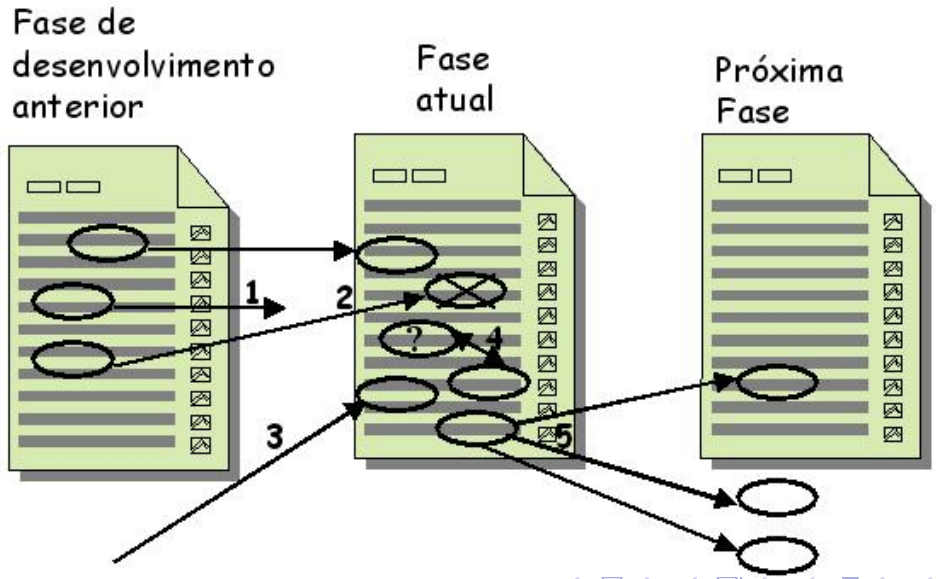
Taxonomia de Defeitos em Documento de Requisitos I

- Utilizada para classificação dos defeitos encontrados.
- Situação ideal:
  - A informação é transformada corretamente.



Taxonomia de Defeitos em Documento de Requisitos II

- Tipos de erros:
  1. A informação é perdida durante a transformação.
  2. A informação é transformada incorretamente.
  3. Informação estranha é introduzida.
  4. A mesma informação é transformada em diversas ocorrências inconsistentes.
  5. A mesma informação possibilita diversas transformações inconsistentes.



Taxonomia de Defeitos em Documento de Requisitos III

- Classes de Defeitos:
  - Defeitos de Omissão
  - Defeitos de Fato Incorreto
  - Defeitos de Inconsitência
  - Defeitos de Ambigüidade
  - Defeitos de Informação Estranha

Defeitos de Omissão I

- Defeito de Omissão (O): qualquer informação necessária que tenha sido omitida.
- Exemplo 1 (Omissão de Funcionalidade). Considere um sistema de biblioteca e os seguintes requisitos funcionais (RF):
  - RF2: O sistema deve solicitar a informação necessária para inserir um item bibliográfico: título, autor, data, lugar, assunto, resumo, número, editor, periódico, congresso.
  - RF3: O sistema deve dar uma mensagem de alerta quando o usuário tentar inserir um item incompleto. Essa mensagem deve questionar o usuário se ele deseja cancelar a operação, completar a informação ou concluir a inserção como está.
    Qual informação é necessária para possibilitar uma inserção incompleta?

Defeitos de Omissão II

- Exemplo 2 (Omissão de Desempenho:)
  - RNF1: O sistema deve fornecer os resultados tão rápido quanto possível.

Tão rápido quanto possível?

Defeito de Fato Incorreto

- Defeito de Fato Incorreto (FI): informação que consta do artefato mas que seja contraditória com o conhecimento que se tem do domínio de aplicação.
- Exemplo: Considere um sistema de empréstimo numa biblioteca e o seguinte RF:
  - RF30: O sistema não deve aceitar devolução de livros se o usuário não estiver com a carteirinha da biblioteca no momento.

Para devolução de livros não é necessário apresentar a carteirinha pois todas as informações já estão registradas no sistema !

Defeito de Inconsistência

- Defeito de Inconsistência (I): informação que consta do artefato mais de uma vez e em cada ocorrência ela é descrita de forma diferente.
- Exemplo: Considere um sistema de empréstimo numa biblioteca e o seguinte RF:
  - RF5: O sistema não deve permitir períodos de empréstimo maiores que 15 dias.
  - RF9: Professores podem retirar livros por um período de 3 semanas.

Qual período deve ser considerado?

Defeito de Ambigüidade

- Defeito de Ambigüidade (A): quando a informação pode levar a múltiplas interpretações.
- Exemplo: Considere um sistema de empréstimo numa biblioteca e o seguinte RF:
  - RF20: Se o número de dias que o usuário está em atraso é menor que uma semana, ele deve pagar uma taxa de R$ 1,00; se o número é maior que uma semana, a taxa é de R$ 0,50 por dia.

Qual a taxa a ser paga se o período for de uma semana?
No primeiro caso, a taxa deve ser calculada por dia?

Defeito de Informação Estranha

- Defeito de Informação Estranha (IE): qualquer informação que, embora relacionada ao domínio, não é necessária para o sistema em questão.
- Exemplo:
  - RF15: Quando um novo livro é adicionado ao acervo, ele permanece em uma prateleira especial por um período de um mês.

Essa informação não é necessária ao sistema !!

Técnica de Leitura Baseada em Perspectiva (PBR) I

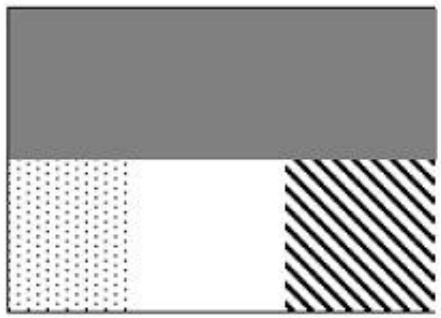
- Técnica de leitura proposta para detectar defeitos em especificações de requisitos.
- Faz com que cada revisor se torne responsável por uma perspectiva em particular.
- Requer que o revisor crie abstrações de um produto de software e responda algumas questões baseado na análise da abstração.
  - As questões a serem respondidas e as regras para criar a abstração são definidas para cada perspectiva diferente.
- Possibilita que o revisor melhore sua experiência em diferentes aspectos do documento de requisitos.
- Assegura que perspectivas importantes sejam contempladas.

Técnica de Leitura Baseada em Perspectiva (PBR) II

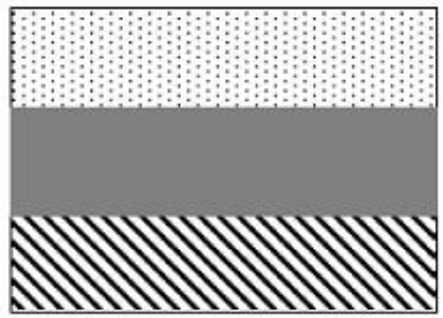
- Cada revisor possui um cenário para guiar seu trabalho de revisão.
- Todo cenário consiste de duas partes:
  - Construir um modelo do documento que está sob revisão a fim de aumentar o entendimento sobre o mesmo.
  - Responder questões sobre o modelo, tendo como foco itens e problemas de interesse da organização.

Técnica de Leitura Baseada em Perspectiva (PBR) III

- Cada revisor vai ler o Documento de Requisitos com olhos diferentes.
- Benefícios:
  - Determina uma responsabilidade específica para cada revisor.
  - Melhora a cobertura de defeitos.



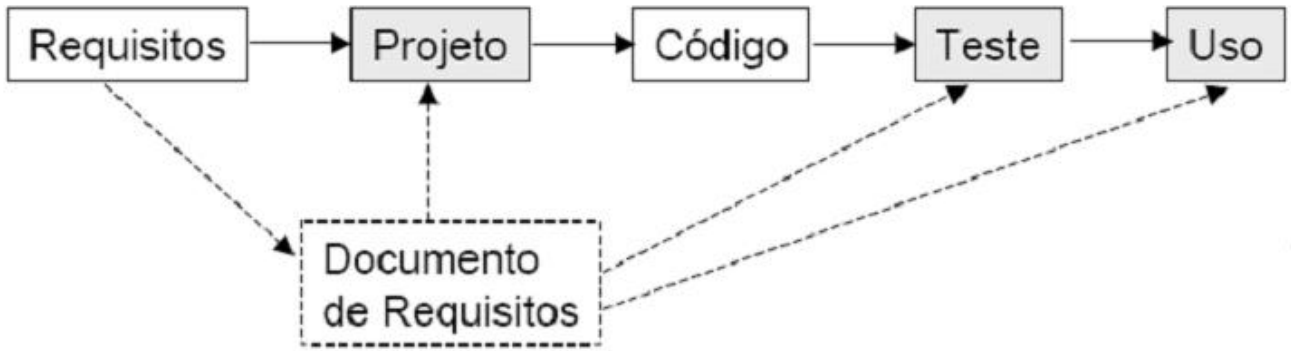
cobertura ad-hoc



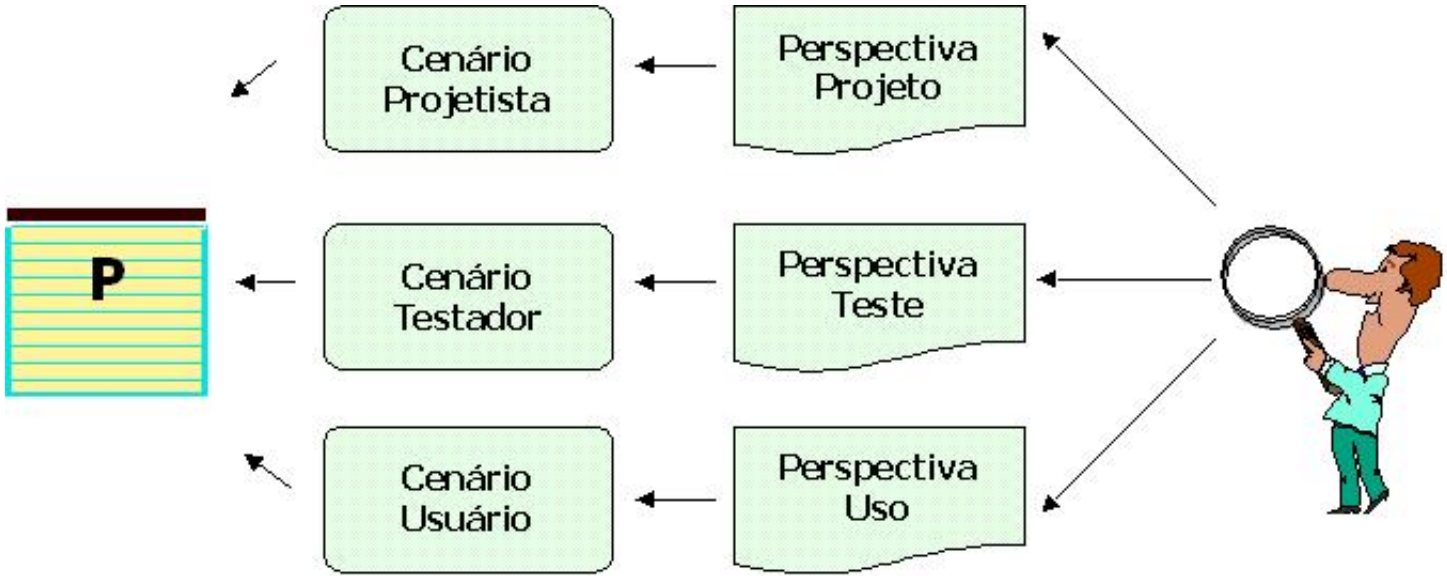
cobertura baseada em perspectiva

Técnica de Leitura Baseada em Perspectiva (PBR) IV

- Várias leituras podem ser feitas no Documento de Requisitos.
  - O projetista que usa o DR para gerar o projeto do sistema.
  - O testador que, com base no DR, deve gerar casos de teste para testar o sistema quando este estiver implementado.
  - O usuário que verifica se o DR está capturando toda funcionalidade que ele deseja para o sistema.



Técnica de Leitura Baseada em Perspectiva (PBR) V



PBR – Visão do Usuário

- Definir um conjunto de funções que o usuário esteja apto a executar.
- Definir o conjunto de entradas necessárias para executar cada função e o conjunto de saídas que são geradas pelas funções.
  - Isso pode ser feito escrevendo todos os cenários operacionais que o sistema deve executar.
- Iniciar com os cenários mais óbvios até chegar nos menos comuns ou condições especiais.
- Ao fazer isso, faça a você mesmo as perguntas a seguir.
  Sugestão: Usar como modelo *Casos de Uso*.

PBR – Visão do Usuário: Questões

- Todas as funções necessárias para escrever os cenários estão especificadas no documento de requisitos ou na especificação funcional?
- As condições para inicializar os cenários estão claras e corretas?
- As interfaces entre as funções estão bem definidas e compatíveis (por ex., as entradas de uma função) têm ligação com as saídas da função anterior?
- Você consegue chegar num estado do sistema que deve ser evitado (por ex., por razões de segurança)?
- Os cenários podem fornecer diferentes respostas dependendo de como a especificação é interpretada?
- A especificação funcional faz sentido de acordo com o que você conhece sobre essa aplicação ou sobre o que foi especificado em uma descrição geral?

PBR – Visão do Testador

- Para cada especificação funcional ou requisito gere um conjunto de casos de teste que assegure que a implementação do sistema satisfaz a especificação funcional ou o requisito.
- Use sua abordagem de teste normal e critérios de teste.
- Ao fazer isso, faça a você mesmo as perguntas a seguir para cada teste.
  Sugestão: Usar como critérios de teste *Particionamento de Equivalência*, *Análise do Valor Limite*.

PBR – Visão do Testador: Questões

- Você tem toda informação necessária para identificar o item a ser testado e o critério de teste?
- Você pode gerar um bom caso de teste para cada item, baseando-se no critério?
- Você tem certeza de que os teste gerados fornecerão os valores corretos nas unidades corretas?
- Existe uma outra interpretação dos requisitos de forma que o programador possa estar se baseando nela?
- Existe um outro requisito para o qual você poderia gerar um caso de teste similar, mas que poderia levar a um resultado contraditório?
- A especificação funcional ou de requisitos faz sentido de acordo com aquilo que você conhece sobre a aplicação ou a partir daquilo que está descrito na especificação geral?

Inspeção em Código-Fonte

- Visa a encontrar defeitos no código-fonte, realizando uma análise estática.
- Vantagens:
  - Programas ficam menos complexos.
  - Subprogramas são escritos em um estilo consistente e obedecem padrões estabelecidos.
  - O desenvolvimento de sistemas torna-se transparente.
  - A estimativa e o planejamento tornam-se mais confiáveis.
  - Facilita a manutenção, com o desenvolvimento de sistemas mais compreensíveis e bem documentados.

Taxonomia de Defeitos em Código

- Duas classificações de defeitos em código:
  - Classificação I:
    - Defeitos de Omissão
    - Defeitos de Comissão
  - Classificação II:
    - Defeitos de Inicialização
    - Defeitos de Computação
    - Defeitos de Controle
    - Defeitos de Interface
    - Defeitos de Dados
    - Defeitos Cosméticos

Taxonomia de Defeitos em Código - I

- Defeitos de Omissão: Esquecimento de algum elemento no programa.
- Exemplo: Falta de um comando que iria atribuir um valor a uma variável.

- Defeitos de Comissão: Segmento de código incorreto.
- Exemplo: Um operador aritmético errado é usado em uma expressão.

Taxonomia de Defeitos em Código - II I

- Defeito de Inicialização: Inicialização incorreta de uma estrutura de dados.
- Exemplo: Atribuir um valor errado a uma variável.

- Defeitos de Computação: Qualquer computação incorreta para geração do valor de uma variável.
- Exemplo: Um operador aritmético errado é usado em uma expressão de atribuição de valor para variável.

Taxonomia de Defeitos em Código - II II

- Defeito de Controle: Causa a execução de um caminho de controle errado para um valor de entrada.
- Exemplo: Predicado incorreto em um comando IF-THEN-ELSE.

- Defeito de Interface: Quando um módulo usa ou faz suposições sobre dados que não fazem parte do seu escopo.
- Exemplo: Passagem de um argumento incorreto para um procedimento, ou quando um módulo assume que irá receber um array inicializado com zero.

Taxonomia de Defeitos em Código - II III

- Defeitos de Dados: Uso incorreto de uma estrutura de dados.
- Exemplo: Determinar incorretamente o último índice de um array.

- Defeitos Cosmético: Erro de escrita no programa.
- Exemplo: Uma mensagem de erro com erro de escrita (ortografia, gramática).

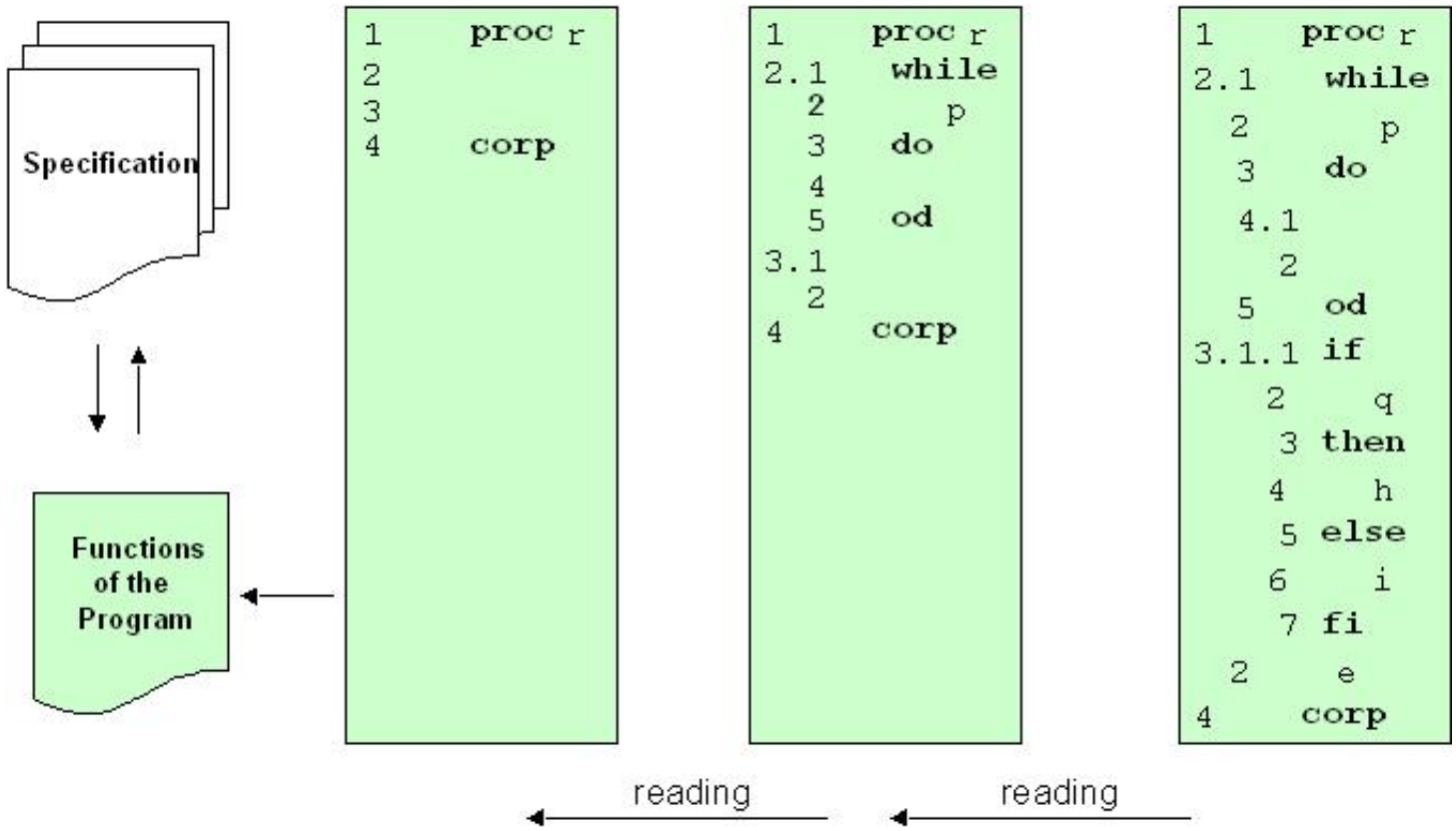
Técnica de Leitura Stepwise Abstraction I

- Técnica de leitura utilizada para detectar defeitos em código-fonte.
  - Code-Reading
- Consiste em desenvolver abstrações funcionais a partir do código-fonte, para determinar a funcionalidade do programa.

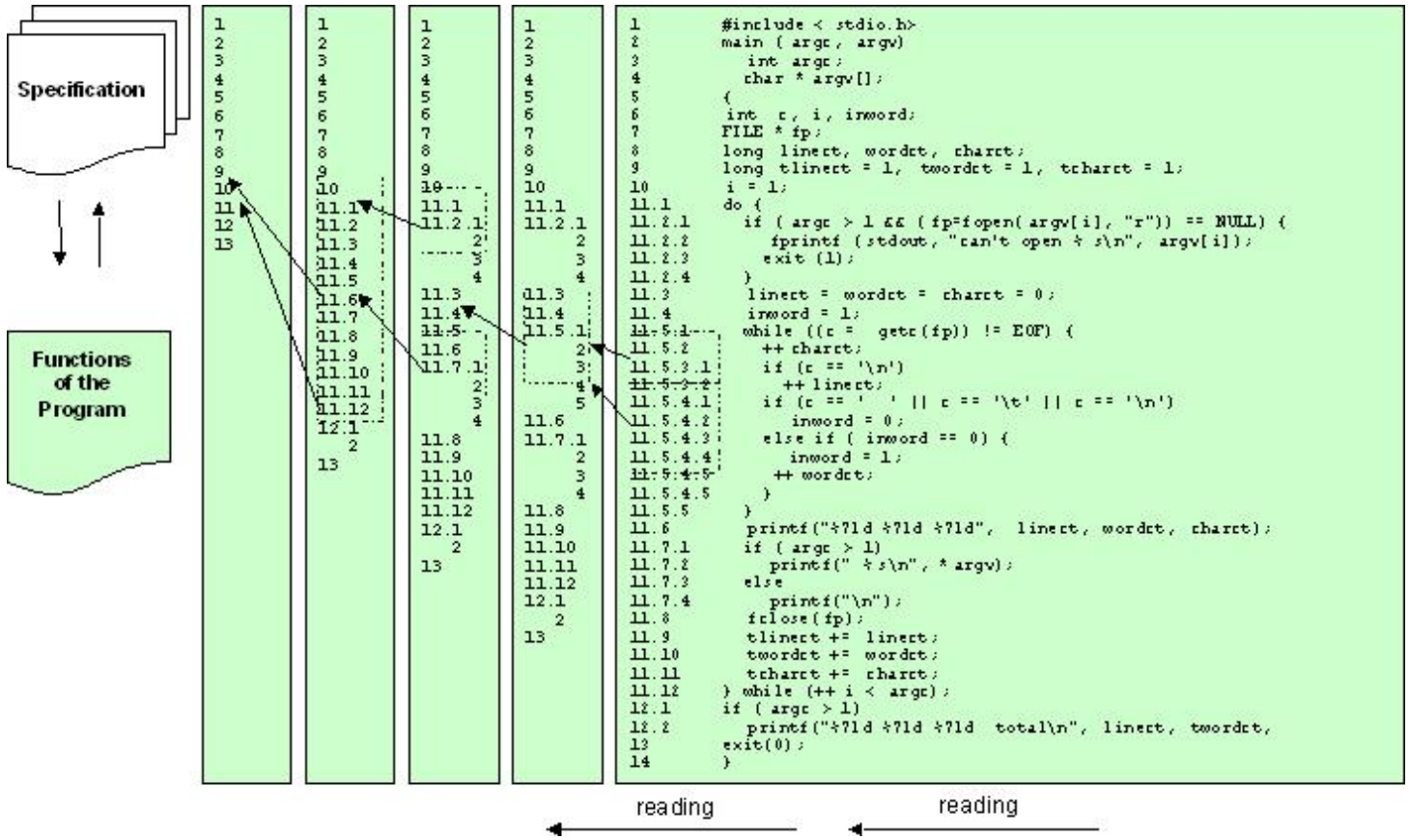
Técnica de Leitura Stepwise Abstraction II

- Os revisores identificam subprogramas no código-fonte e escrevem suas próprias especificações para os subprogramas (abstrações de funcionalidade).
  - As abstrações construídas devem ser combinadas em uma abstração mais geral, até que se tenha capturado a função completa do programa.
- Inconsistências são detectadas comparando a especificação original com a especificação construída por meio das abstrações.

Técnica de Leitura Stepwise Abstraction III



Técnica de Leitura Stepwise Abstraction IV



Técnica de Leitura Stepwise Abstraction V

- Exemplo de abstrações de código:

```
01: if x != 0 then
02:     y := 5;
03: else
04:     z := z - x;
05: endif
06: if z > 1 then
07:     z := z / x;
08: else
09:     z := 0;
10: endif
```

Abstrações linhas 01-05:

```
if x <> 0 then y:=5 else z:=z-x
```

Abstrações linhas 06-10:

```
if z > 1 then z:=z/x else z:=0
```

Abstrações linhas 01-10:

```
if x <> 0 then
  if z > 1
       then y:=5,z:=z/x
       else y:=5,z:=0
   else if z > 1
       then z:=z-x, z:=z/x
       else z:=z-x, z:=0
```

Princípios para o Processo de Leitura de Código I

1. Detectar como o programa irá falhar durante a execução.
   - Construir as abstrações.
     - Determinar o comportamento de um componente através da leitura.
     - Determinar as dependências entre as funções individuais no código. Normalmente, as funções estão ordenadas no código, de modo que as funções *low-level* (folhas) estão no ínicio do código e as funções *high-level* (root) estão no final. Inicie aplicando a técnica de leitura de código nas funções *folha* caminhando em direção à *raiz*.
     - Desenvolver um entendimento da estrutura de cada função individual, identificando as estruturas elementares (seqüência, condições, atribuições, repetições).

Princípios para o Processo de Leitura de Código II

- Marcar as estruturas identificadas, desenhando quadrados para delimitá-las.
- Combinar essas estruturas em outras mais externas até construir um quadrado que representa uma única função.
- Agregar as funções elementares de acordo com o fluxo de controle do programa.

- Isolar as falhas.
  - Comparar as abstrações com a especificação para detectar possíveis falhas.

2. Isolar os defeitos que levariam a falhas.
   - Buscar as causas do comportamento inadequado (defeitos) do programa.
   - O entendimento do programa adquirido durante a leitura facilita a localização do defeito.

Concluindo...

- Inspeção não requer necessariamente a execução do sistema e assim pode ser usada antes da implementação estar concluída.
  - Pode ser aplicada em qualquer representação do sistema (requisitos, projeto, dados de testes, ...)
- Quando um erro é encontrado durante a inspeção, é conhecida também sua natureza e localização.
  - Isso não ocorre com Teste de Software.
- Inspeção e Teste são técnicas de V&V complementares. Ambas devem ser usadas!!!
  - Inspeções podem checar a conformidade com especificação mas não a conformidade com os requisitos reais do cliente!
  - Inspeções não podem checar características não funcionais como desempenho, usabilidade....

Exercício de Fixação